Urządzenia Peryferyjne Laboratorium

15.01.2024

Grupa nr 7B, Czwartek, godz. 11:00, tydzień nieparzysty

Autorzy:

264488 Adam Czekalski

252560 Karol Szydlak

nr	Treść zadania	data wykonania
6.	Ćwiczenie 14 Transmisja WiFi.	21/12/2023

1. Wstęp teoretyczny

Standardy Wi-Fi – zestawy reguł i protokołów, które określają sposób w jaki urządzenia komunikują się bezprzewodowo w sieci. Najczęściej stosowany standard w urządzeniach to IEEE 802.11:

- 802.11 – jest to pierwszy standard opublikowany w 1997 roku. Określa 2 możliwe prędkości transmisji: 1Mb/s oraz 2Mb/s. Pozwala na pracę w pasmie 2.4GHz.
- 802.11b – opublikowany w 1999 roku. Pozwala na prędkość do 11Mb/s i pozwala na pracę niezmiennie w pasmie 2.4GHz.
- 802.11a – opublikowany w 1999 roku, stosowany w urządzeniach od 2001 roku. Pracuje w pasmie 5GHz, pozwala na prędkość do 54Mb/s, w praktyce jednak stablina prędkość to 20Mb/s.
- 802.11g – opublikowany w czerwcu 2003 roku, pracuje nadal w pasmie 2.4GHz, ale pozwala na transfer z prędkością do 54Mb/s. Posiadał wsteczną kompatybilność ze standardem 802.11b, jednak dla starszych urządzeń redukował prędkość do założeń 802.11b - 11Mb/s
- 802.11n – zatwierdzony we wrześniu 2009 roku, w teorii oferował transfer do 600Mb/s w pasmie nadawania o szerokości 20MHz, w praktyce jednak do 150Mb/s w pasmie nadawania o szerokości 40MHz. Po raz pierwszy umożliwiono wykorzystanie wielu anten do nadawania i odbierania sygnału oraz potrafią wykorzystać 2 kanały transmisyjne żeby stworzyć jedno połączenie, co umożliwia pracę w 2 pasmach na raz – 2.4 GHz oraz 5GHz. Pasmo 2.4GHz charakteryzuje się większą odpornością na zakłócenia i większym zasięgiem, kosztem prędkości; natomiast pasmo 5GHz jest szybsze kosztem mniejszej odporności na zakłócenia i zasięgiem
- 802.11ac – zatwierdzone w styczniu 2014 roku, zapewnia pracę w pasmie 5GHz, pozwala osiągać prędkości do 1Gb/s przy zastosowaniu wielu stacji, przy jednej natomiast do 500Mb/s
- 802.11ax – najnowszy standard, wprowadzony w 2019 roku, pozwala na pracę w pasmie 2.4GHz oraz 5GHz. Przyspieszono prędkość w bardziej zatłoczonych miejscach. Pozwala na osiągnięcie maksymalnie 10Gb/s.

Warstwy fizyczne:

- DSSS (ang. Direct Sequence Spread Spectrum) – bezpośrednie modulowanie nośnej sekwencją kodową, technika rozpraszania widma w systemach szerokopasmowych
- FHSS (ang. Frequency Hopping Spread Spectrum) - „skakanie” sygnału po częstotliwościach w kolejnych odstępach czasu, w dostępnym widmie
- OFDM (ang. Orthogonal Frequency-Division Multiplexing) - metoda zwielokrotnienia w dziedzinie częstotliwości polegająca na jednoczesnej transmisji wielu strumieni danych na ortogonalnych częstotliwościach nośnych

Kanały:

W routerze następuje podział na maksymalnie 14 kanałów o szerokości 22MHz każdy. Wszystkie, oprócz 3 kanałów, nachodzą na siebie. W Polsce można wykorzystywać pasmo od 2400,0 do 2483,5 MHz, czyli od kanału 1 do 13.

Zasady realizacji transmisji Wi-Fi:

- Router – urządzenie przekazujące pakiety danych między różnymi sieciami. Oprócz tego, korzysta z adresów IP, aby kierować pakiety do właściwych interfejsów
- Access Point (w skrócie: AP) – urządzenie pozwalające na zwiększenie zasięgu sieci Wi-Fi

- Klient – urządzenie korzystające z usług świadczonych przez inne urządzenie (serwer)
- Serwer – urządzenie dostarczające usługi innym urządzeniom (klientom)
- Station – stacja robocza, czyli urządzenie którego używa użytkownik końcowy
- UDP – protokół znajdujący się w warstwie 3. TCP/IP (transportowej). Jest bezpołączeniowy, nie posiada mechanizmów kontroli przepływu i retransmisji. Pozwala to na szybszą transmisję danych. Stosowany jest w wideokonferencjach, grach sieciowych. Nie gwarantuje dostarczenia datagramu
- TCP – także znajduje się w warstwie 3., ale jest połączeniowy i posiada mechanizmy kontroli przepływu. Gwarantuje dostarczenie wszystkich pakietów w całości w ustalonej kolejności, kosztem prędkości.
- WiFi – technologia bezprzewodowej komunikacji, która pozwala na korzystanie z internetu bez użycia okablowania
- P2P – rodzaj komunikacji (peer-to-peer), który zapewnia każdemu hostowi takie same uprawnienia

Środowiska/układy wspierające transmisje WiFi:

- Android – system operacyjny Android posiada wbudowane API do obsługi transmisji Wi-Fi. Programista może zarządzać połączeniami Wi-Fi, skanować sieć oraz tworzyć aplikacje z pomocą np. Android SDK i odpowiedniego frameworka
- ESP01 oraz ESP32 – mikrokontrolery posiadające wbudowaną obsługę Wi-Fi, można je zaprogramować za pomocą Arduino IDE

Biblioteki wspierające oprogramowywanie transmisji Wi-Fi:

- Moduł os – pozwala na wydawanie poleceń z linii komend
- Moduł socket – pozwala na komunikację sieciową między różnymi urządzeniami lub procesami w ramach sieci komputerowej

2. Kod

W ramach zajęć napisano aplikację umożliwiającą:

- Aktywację i wyłączenie karty sieciowej Wi-Fi
- Połączenie z wybranym urządzeniem Wi-Fi
- Transmisję danych przez Wi-Fi za pomocą gniazd TCP

Program został napisany w języku Python.

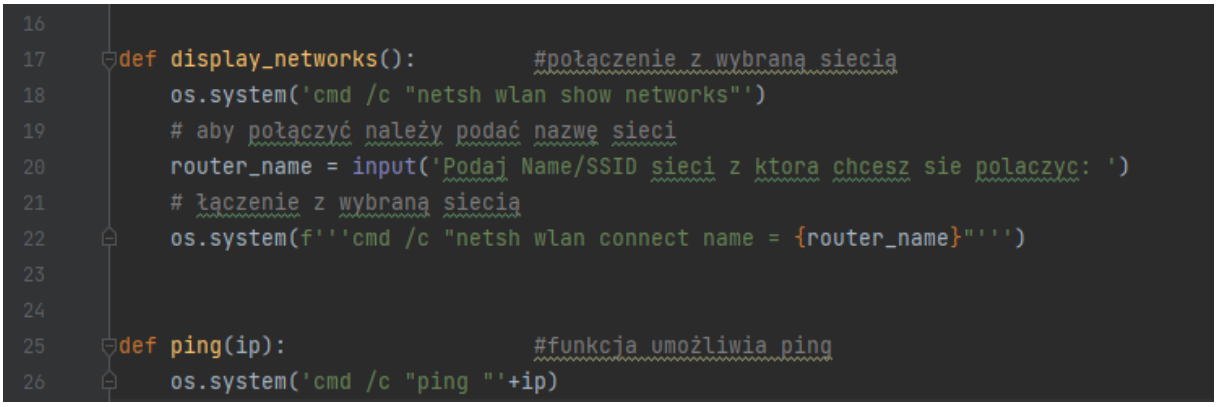
2.1 Włączenie i wyłączenie urządzenia

Włączenie i wyłączenie urządzenia odbywa się z użyciem biblioteki os pozwalającej na wydawanie poleceń z linii komend. Na poniższych rysunkach przedstawiono metody realizujące omawianą funkcjonalność.

Rysunek 1. Włączenie oraz wyłączenie urządzenia.

2.2 Połączenie z wybranym urządzeniem Wi-Fi

Podobnie jak w punkcie 2.1, połączenie z wybranym urządzeniem wykorzystuje bibliotekę os, metodę przedstawia poniższy rysunek.



Rysunek 2. Połączenie z wybraną siecią Wi-Fi

Metoda *display_network()* wyświetla dostępne sieci Wi-Fi, po podaniu przez użytkownika nazwy sieci łączy się z siecią o zadanej nazwie. Na rysunku przedstawiono także metodą o nazwie *ping()*, która umożliwia pingi między urządzeniami w sieci celem weryfikacji poprawności połączenia.

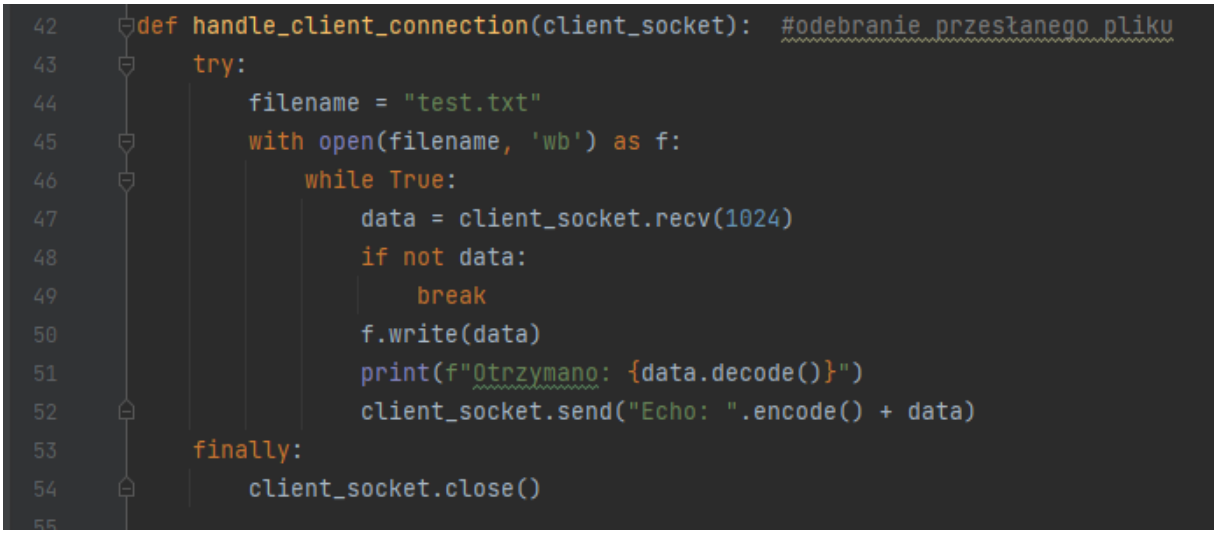
2.3 Transmisja danych

Do transmisji danych użyto gniazd TCP. Przy uruchomieniu programu aktywowany jest wątek odpowiedzialny za obsługa przychodzących połączeń. Przy przesyłaniu plików użytkownik podaje nazwę wysyłanego pliku, oraz adres IP adresata.

```
def start_server(host, port=50000):        #metoda akceptująca połączenie oraz tworząca wątek obsługujący połączenie z klientem
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(5)
    print(f"Serwer nasłuchuje na {host}:{port}")
    def listen(server):
        while True:
            client_sock, address = server.accept()
            print(f"Akceptacja połączenia od {address}")
            client_handler = threading.Thread(
                target=handle_client_connection,
                args=(client_sock,)
            )
            client_handler.start()
    t = threading.Thread(target=listen, args = (server,))
    t.start()



def run_Server():               #metoda uruchamia wątek nasłuchujący na danym porcie, który akceptuje połączenia
    my_ip = get_local_ip()
    server_thread = threading.Thread(target=start_server(my_ip,50000,))
    server_thread.start()

```

Rysunek 3. Metody obsługujące przychodzące połączenia.

Metoda *start_server()* w osobnym wątku nasłuchuje na porcie 50000, akceptuje przychodzące połączenia tworząc nowy wątek obsługujący połączenie klienta nazwany *client_handler()* przedstawiony na poniższym rysunku:



```
def handle_client_connection(client_socket):  #odebranie przesłanego pliku
    try:
        filename = "test.txt"
        with open(filename, 'wb') as f:
            while True:
                data = client_socket.recv(1024)
                if not data:
                    break
                f.write(data)
                print(f"Otrzymano: {data.decode()}")
                client_socket.send("Echo: ".encode() + data)
    finally:
        client_socket.close()
```

Rysunek 4. Metoda obsługująca odbieranie wysyłanych plików.

Przesyłane dane są zapisywane w pliku tekstowym o nazwie *test.txt*, po czym utworzone gniazdo jest zamykane.

Poniżej przedstawiono kod odpowiedzialny za wysłanie danych:

```python
def send_file(filename, host='hostname', port=50000):   #przesłanie pliku
    s = socket.socket()
    s.connect((host, port))

    with open(filename, 'rb') as f:
        data = f.read(1024)
        while data:
            s.send(data)
            data = f.read(1024)

    s.close()
    print("Plik został wysłany.")
```

Rysunek 5. Metoda odpowiedzialna za wysyłanie danych.

2.4 Menu

W programie zaimplementowano menu, umożliwiające interakcję z użytkownikiem:

```python
def main():
    my_ip = get_local_ip()
    start_server(my_ip)
    option = 0
    while True:

        print("1.Connect to Wi-Fi network")
        print("2.Send file")
        print("3.Ping")
        print("4.Wlacz WiFi")
        print("5.Wylacz Wifi")
        print("0.Wyjdz")
        option = input("Enter option\n")


        if option.__eq__("1"):
            display_networks()

        elif option.__eq__("2"):
            filename = input("Give file name")
            hostname = input("Give hostname")
            send_file(filename, hostname)

        elif option.__eq__("0"):
            return
        elif option.__eq__("3"):
            ip = input("Give ip:")
            ping(ip)
        elif option.__eq__("4"):
            enable_wifi_card()
        elif option.__eq__("5"):
            disable_wifi_card()
```

Rysunek 6. Menu

3. Źródła

- https://pl.wikipedia.org/wiki/IEEE_802.11
- https://www.netia.pl/pl/blog/standardy-wi-fi-802-11-a-b-g-n-ac-ax
- https://www.dipol.com.pl/warstwy_fizyczne_w_standardzie_ieee_802_11_bib511.htm
- https://pl.wikipedia.org/wiki/Klient_(informatyka)
- https://pl.wikipedia.org/wiki/Serwer
- https://pl.wikipedia.org/wiki/User_Datagram_Protocol
- https://pl.wikipedia.org/wiki/Protok%C3%B3%C5%82_sterowania_transmisj%C4%85
- https://pl.wikipedia.org/wiki/Peer-to-peer
- https://pl.wikipedia.org/wiki/OFDM
- https://pl.wikipedia.org/wiki/DSSS
- https://pl.wikipedia.org/wiki/FHSS